Module Guide for RoCam

Team #3, SpaceY
Zifan Si
Jianqing Liu
Mike Chen
Xiaotian Lou

November 4, 2025

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
RoCam	Explanation of program name
UC	Unlikely Change

Contents

List of Tables

List of Figures

3 Introduction

Decomposing a system into modules awa is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility, and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The computer vision model. Frequent changes to the vision model might be required to adapt the system to new deployment envirouments.

AC2: The user interface. The user interface may need to be updated based on the feedback of our customers during the field test.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: The hardware platform. The system relys hardware acceleration for running the computer vision model, encoding, and video output. Changing the hardware platform may require refactoring all the related algorithms.

UC2: The tracked object. The system is designed to track only rockets. We are able to optimize the performance of the system based on deterministic movement patterns of a rocket. Changing the tracked object to something other than a rocket voids this optimization.

UC3: Fundamental changes to the use case. The system is designed to be operated remotely by a human operator.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Jetson Module

M2: Gimbal Abstraction Module

M3: Computer Vision Module

M4: Tracking Module

M5: Output Video Module

M6: Recording Module

M7: State Management Module

M8: API Gateway Module

M9: UI Module

M10: Preview Module

M11: Manual Control Module

M12: Recording Management Module

M13: Configuration Module

Level 1	**Level 2**
Jetson Module	Gimbal Abstraction Module
	Computer Vision Module
	Tracking Module
	Output Video Module
	Recording Module
	State Management Module
	API Gateway Module
UI Module	Preview Module
	Manual Control Module
	Recording Management Module
	Configuration Module

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *RoCam* means the module will be implemented by the RoCam software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Jetson Module (M1)

The jetson module includes all the code that will be running on the jetson.

7.1.1 Gimbal Abstraction Module (M2)

Secrets: The communication protocol of the gimbal

Services: controls the gimbal and get information from the gimbal

Implemented By: RoCam

Type of Module: Library

7.1.2 Computer Vision Module (M3)

Secrets: The computer vision algorithms

Services: Detects rockets from the camera feed in real time.

Implemented By: RoCam

Type of Module: Library

7.1.3 Tracking Module (M4)

Secrets: The tracking algorithms

Services: Calculates the angle of the gimbal to keep the rocket in the frame.

Implemented By: RoCam

Type of Module: Library

7.1.4 Output Video Module (M5)

Secrets: The algorithm to efficiently compose the video feed

Services: Outputs the video feed to the display with camera feed and information overlay.

Implemented By: RoCam

Type of Module: Library

7.1.5 Recording Module (M6)

Secrets: The algorithm to encode the video feed and log files/

Services: Records the video feed and log files to the disk.

Implemented By: RoCam

Type of Module: Library

7.1.6 State Management Module (M7)

Secrets: The data structure to store the state of the system

Services: Manages the state transitions of the system.

Implemented By: RoCam

Type of Module: Record

7.1.7 API Gateway Module (M8)

Secrets: The API endpoints and their corresponding logic.

Services: Provides the API endpoints for the UI.

Implemented By: RoCam

Type of Module: Library

7.2 UI Module (M9)

The UI module includes all the code that will be used for the operator to interact with the system.

7.2.1 Preview Module (M10)

Secrets: How to retrieve and display the video feed from the API Gateway.

Services: Displays preview of the video feed to the user.

Implemented By: RoCam

Type of Module: Library

7.2.2 Manual Control Module (M11)

Secrets: Intuitive user interface for manual control, and how to send manual control commands to the API Gateway.

Services: Allows the operator to control the gimbal manually via a user interface.

Implemented By: RoCam

Type of Module: Library

7.2.3 Recording Management Module (M12)

Secrets: How to list, delete, and download the recorded videos and log files.

Services: Allows the operator to list, delete, and download the recorded videos and log files.

Implemented By: RoCam

Type of Module: Library

7.2.4 Configuration Module (M13)

Secrets: The UI for configuring the system.

Services: Allows the operator to configure the system via a user interface.

Implemented By: RoCam

Type of Module: Library

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
FR-1	M1, M5, M3
FR-2	M2, M4, M11, M8
FR-3	M3, M1
FR-4	M7, M2, M3
FR-5	M7, M3, M4
FR-6	M11, M8, M2, M7
FR-7	M4, M3, M2, M7
FR-8	M5, M1, M8, M9, M10
FR-9	M5, M2, M9, M8, M10
FR-10	M10, M9, M7, M8, M5
FR-11	M6, M12, M8, M9, M10, M5
FR-12	M12, M8, M6, M9,

Table 2: Trace Between Functional Requirements and Modules

Req.	Modules
AR-1	M9, M12, M13
AR-2	M9, M12
SR-1	M9, M13

Table 3: Trace Between Appearance and Style Requirements and Modules

Req.	Modules
UHR-EZ-1	M9
UHR-EZ-2	M9, M8
UHR-EZ-3	M9, M8, M13
UHR-EZ-4	M9, M13, M8, M7
UHR-PI-1	M9, M13, M10, M12
UHR-PI-2	M9, M13
UHR-LR-1	M9
UHR-UPR-1	M9
UHR-UPR-2	M9, M8
UHR-UPR-3	M9
UHR-AR-1	M9, M7, M8

Table 4: Trace Usability and Humanity Requirements and Modules

Req.	Modules
PR-SLR-1	M1, M3, M4, M2, M7
PR-SLR-2	M1, M3, M5, M4
PR-SLR-3	M5, M1, M8
PR-SCR-1	M9, M11, M7, M8, M2
PR-SCR-2	M4, M7, M3, M2, M1
PR-SCR-3	M9, M8, M7, M1
PR-SCR-4	M9, M7, M8, M2
PR-PAR-1	M4, M3, M2, M5, M8, M1
PR-RFR-1	M9, M8, M7
PR-RFR-2	M7, M4, M2, M8
PR-RFR-3	M1, M8, M9, M7, M1
PR-RFR-4	M4, M3, M7, M2, M1
PR-RFR-5	M4, M3, M2, M7
PR-CR-1	M6, M12, M1

Table 5: Trace Between Performance Requirements and Modules

Req.	Modules
OER-EPE-1	M1, M2
OER-INT-1	M5, M1
OER-INT-2	M2, M8, M1
OER-RR-1	M6, M12, M1

Table 6: Trace Between Operational and Environmental Requirements and Modules

Req.	Modules
SEC-IR-1	M9, M11, M8, M7, M2, M1, M4
SEC-AUR-1	M6, M12, M8, M7, M9

Table 7: Trace Between Security Requirements and Modules

Req.	Modules
CUL-CR-1	M9, M10, M13
CUL-CR-2	M9, M10, M13

Table 8: Trace Between Cultural Requirements and Modules

Req.	Modules
COM-LR-1	M9, M6, M12, M8, M1

Table 9: Trace Between Compliance Requirements and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 3 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

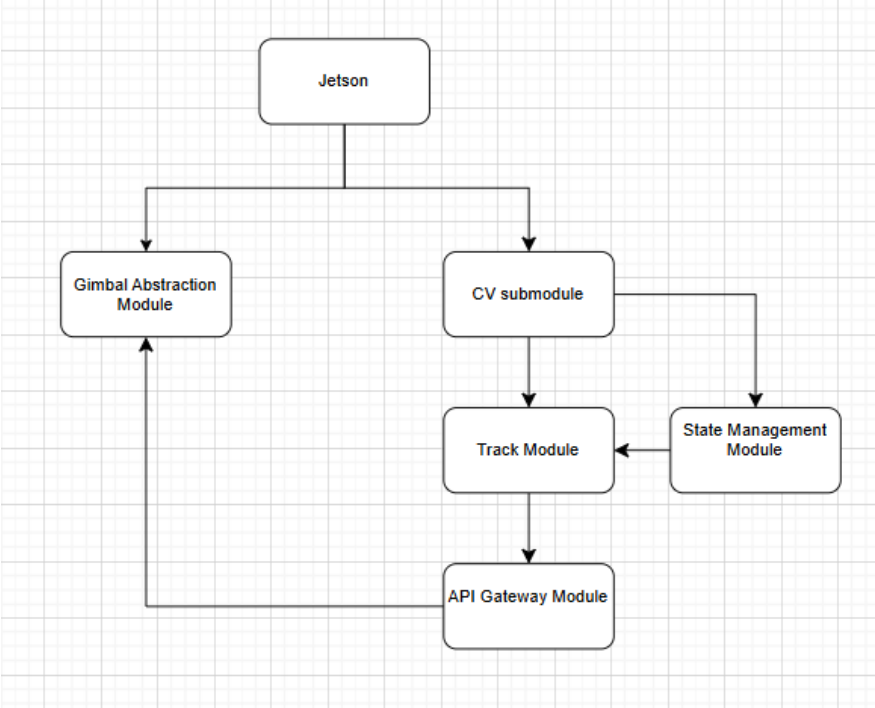

Figure 1: Use hierarchy among backend modules

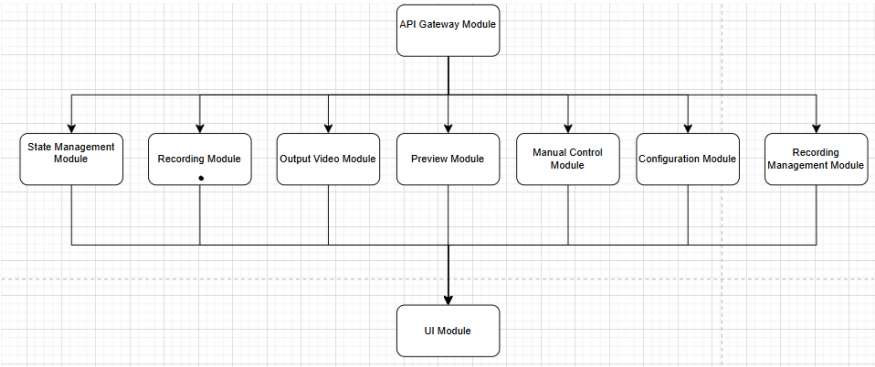

Figure 2: Use hierarchy among frontend modules

Figure 3: Use hierarchy among modules

10 User Interfaces

11 Design of Communication Protocols

12 Timeline

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.